Proceedings of the 1996 IEEE
International Conference on Robotics and Automation
Minneapolis, Minnesota - April 1996

Speeding Up Interference Detection Between Polyhedra *

Pablo Jiménez Carme Torras

Institut de Cibernètica (CSIC - UPC)
Diagonal 647, 2 planta
08028 Barcelona, SPAIN

Abstract

A classical paradigm for interference detection between polyhedra consists in testing all edges of one polyhedron against all faces of the other one for intersection. If the relative orientation of the polyhedra is fixed, only certain edge-face pairs can intersect first, when the polyhedra come into contact. These candidate pairs are efficiently determined using a representation which we call Spherical Face Orientation Graph. By applying the interference test to candidates only, the computational effort is significantly reduced, as shown by experimental results with convex polyhedra. In the non-convex case, the strategy is conservative, but it still leads to savings.

1 Introduction

Interference detection between polyhedra is a central issue in the context of collision detection and robot motion planning. The obvious way of performing interference detection, if the polyhedra are described using a boundary representation, is to decompose the problem into elementary tests involving the boundary primitives. If the complexity of the polyhedra is high, procedures have to be devised that avoid having to perform every elementary test. In this direction, hierarchical representations may save a great amount of computational work, if the interference situation can be decided at the first levels of the hierarchy or if its refinement can be restricted to the area where interference is most likely to occur. Of course, a preprocessing step is necessary in order to obtain the hierarchical representation. Optimal $O(\log n \log m)$ solutions exist for preprocessed convex polyhedra having n and m edges [1], preprocessing requiring linear time. There also exists an $O((n + m + s)\log(n + m + s))$ *intersection computation* algorithm between two preprocessed polyhedra, one of which is convex [2], where s is the number of edges of the intersection polyhedron. Both methods, if applied to non-convex polyhedra, require a previous step of decomposition of the polyhedra into convex entities, since the hierarchical representation exploits convexity. Although there exist very efficient decomposition techniques [3], there are also many situations where a large number of new faces will be created by decomposition. Therefore, this previous step may lead to a large increment in the global complexity.

Thomas and Torras [4] have proposed an algorithm for solving the interference detection problem between non-convex polyhedra without decomposing them. It is based on an edge-face intersection test, following the line established in [5], which combines predicates associated to basic contact functions [6] in a manner similar to Canny's *disjunctive form* [7], but being also valid for non-convex faces.

Although the worst-case complexity of the algorithm is necessarily quadratic (all edges of one polyhedron may have to be tested against all faces of the other), the aim of the research described in this paper is to lower the computational cost as much as the particular situation permits. The approach that has been followed can be viewed as an alternative to the methods based on hierarchical representations, as far as the goal is also to restrict the number of elementary tests to perform, but the key geometric issue behind it is different: applicability instead of proximity. While proximity has been applied in the algorithms that determine interference by computing the distance between the polyhedra, as in [1,8], less use has been made of the applicability concept. Applicability has only been implicitly used in [9] for restricting the search space in an incremental distance

*This research has been partially supported by the ESPRIT III Basic Research Actions Program of the EC under contract No. 6546 (project PROMotion).

0-7803-2988-4/96 $4.00 © 1996 IEEE

computation algorithm between convex polyhedra.

The techniques developed here work for all kinds of polyhedra where for every vertex the following condition holds: *The faces adjacent to the vertex form a simple circuit* (therefore, two pyramids joined by their apices would be considered as two different polyhedra, as the condition does not hold for the common vertex if the whole is considered as a single polyhedron). An additional requirement is that the polyhedra have to be connected regularized sets. Therefore faces may be convex as well as non-convex polygons, edges may be convex or concave, depending on their dihedral angle, and vertices may be convex (all adjacent edges are convex), concave (all adjacent edges are concave), or mixed (there are convex as well as concave adjacent edges).

This paper is structured as follows: The first two sections are devoted to previous results that are needed for a clear understanding of the main contributions of this paper, described in Sections 4, 5, and 6. In Section 2, a brief description of Thomas and Torras' elementary edge - face intersection test is given. In the next section, the concept of *applicability* of a contact is described, as well as the way it can be used to restrict the set of candidates to undergo elementary edge-face intersection tests. A new spherical representation of polyhedra is introduced in Section 4, that allows to efficiently determine these candidates in the convex as well as in the non-convex case. Section 5 is devoted to the algorithms that determine the set of candidates to be considered: a general algorithm is presented, and also a specific algorithm for the particular case where the polyhedra are convex, which has already been implemented and optimal results have been obtained. Finally, some conclusions are given, as well as some possible lines of further research, mainly oriented towards a widening of the scope of applications of the basic interference detection method towards more general collision detection algorithms.

2 The edge - face intersection test

In what follows, vertices, edges, and faces are referred to by position v, direction e, and normal f vectors, respectively. The fundamental edge (e) - face (f) test consists in determining whether edge e intersects or not face f, which can be non-convex. If intersection actually occurs, two conditions must simultaneously hold:

- both extremes of the edge, $\partial^+ e$ and $\partial^- e$, must be in opposite halfspaces, of those defined by the plane that contains the face f, and
- the line supporting the edge e must intersect the face f.

The basic predicate $\mathbf{A}_{v,f}$ indicates in which halfspace of those defined by the plane that supports face f lies vertex v. The predicate is true if this halfspace is the same where the normal vector of the plane is pointing to, negative otherwise. This truth value corresponds to the sign of the contact function between the vertex and the face, which can be calculated as a determinant involving the coordinates of three points on the face and the coordinates of the vertex. If the two predicates $\mathbf{A}_{\partial^+ e,f}$ and $\mathbf{A}_{\partial^- e,f}$ have different truth values, the first condition for edge-face piercing will be met. Therefore, the two predicates have to be combined through the *exclusive OR* (XOR, denoted by $\oplus$) operator.

As for the second condition, if a plane f_e containing edge e is constructed, the number of edges e_f of f (i.e., $e_f \in \partial f$) piercing one of the halfplanes of f_e have to be considered, where these halfplanes are defined by the line supporting edge e. If the number of edges piercing any one of these halfplanes is odd, the line that supports e intersects the face. Note that the face may be non-convex. It can be shown [4] that this condition holds iff

$$\bigoplus_{e_f \in \partial f} (\mathbf{A}_{\partial^+ e_f, f_e} \oplus \mathbf{A}_{\partial^- e_f, f_e}) \wedge (\mathbf{A}_{\partial^- e_f, f_e} \oplus \mathbf{B}_{e,e_f}) \quad (1)$$

is true. Here, a second type of basic predicate has been introduced, $\mathbf{B}_{e,e_f}$, which is true iff $\langle e \times e_f, \partial^+ e - \partial^- e_f \rangle$ is positive (where $\langle \cdot, \cdot \rangle$ stands for the inner product), that is, its truth value depends on the relative orientation of e and e_f. The first part of this formula is true if the edge e_f is piercing plane f_e. The second part ensures that only the edges piercing one of the halfplanes are taken into account.

Thus, the edge-face intersection test is based on the truth value of the following composite predicate:

$$(\mathbf{A}_{\partial^+ e,f} \oplus \mathbf{A}_{\partial^- e,f}) \wedge \left[\bigoplus_{e_f \in \partial f} (\mathbf{A}_{\partial^+ e_f, f_e} \oplus \mathbf{A}_{\partial^- e_f, f_e}) \wedge (\mathbf{A}_{\partial^- e_f, f_e} \oplus \mathbf{B}_{e,e_f})\right] \quad (2)$$

To perform interference detection between the boundaries of two polyhedra, this formula has to be applied to all possible edge - face pairings. An AND-OR-XOR tree represents the search space, where the root is an OR node: it suffices that just one subtree corresponding to a particular edge - face combination (i.e., to Equation 2) be true in order to report that an interference has been detected.

The situation in which there is no boundary intersection, because one polyhedron is completely inside the other (and therefore no edge-face intersection actually occurs), can also be handled by using the same basic predicates [4]. The idea is to perform an edge-face intersection test, where the "edge" consists in a segment drawn from an arbitrary vertex of one polyhedron, say **P**, to "infinity" (any point which is far enough). If **P** is inside the other polyhedron, **Q**, an odd number of faces of **Q** will be pierced by this straight halfline. The same test must be performed reversing **P** and **Q**.

3 Applicability conditions and geometric pruning

When two initially non-intersecting polyhedra undergo an arbitrary relative translational motion with respect to one another, only certain edge-face intersections can occur first. In the context of Motion Planning in Robotics it is often enough to determine these first intersections. This is the aim of our geometric pruning techniques.

These edge-face candidates can be easily obtained using the *applicability constraints*, developed in [10] for the case in which the polyhedra are convex. The applicability constraints allow one to determine the basic vertex - face and edge - edge contacts[1] that are possible between two polyhedra whose relative orientation remains fixed but which are allowed to translate. The applicability constraints can be expressed as follows [10]:

Type A contact For a given relative orientation between two polyhedra, the contact between a vertex v of one polyhedron and a face f of another polyhedron is applicable iff $\forall v_i$ adjacent to v, $\langle v_i, f\rangle - \langle v, f\rangle \geq 0$.

Type B contact For a given relative orientation between two polyhedra, the contact between an edge e_m of one polyhedron and an edge e_n of another polyhedron is applicable iff $k_a \neq k_b$, where $k_a = \text{sign}(\langle T_1, f_p\rangle) = \text{sign}(\langle T_2, f_p\rangle)$, and $k_b = \text{sign}(\langle T_3, f_p\rangle) = \text{sign}(\langle T_4, f_p\rangle)$, with $T_i = s_i \cdot (f_i \times e_m), f_i$ adjacent to $e_m, T_j = s_j \cdot (f_j \times e_n), f_j$ adjacent to $e_n; s_i, s_j \in \{+1, -1\} | T_l$ is oriented towards the interior of face f_l (see Figure 1), and $f_p = e_m \times e_n$.

If the contact between a vertex and a face is applicable, only one of the edges adjacent to the vertex has to be considered as candidate for intersection with the face. In the general case, where the situation is not of parallel faces or an edge parallel to a face, only one vertex will be applicable with respect to a given face. Therefore, no more edges will have to be considered as candidates for intersection with this face, following this criterion. In a similar way, if the contact between two edges is applicable, the candidate pairings to be considered are formed by each one of the edges and the adjacent faces to the other edge (see Figure 1).

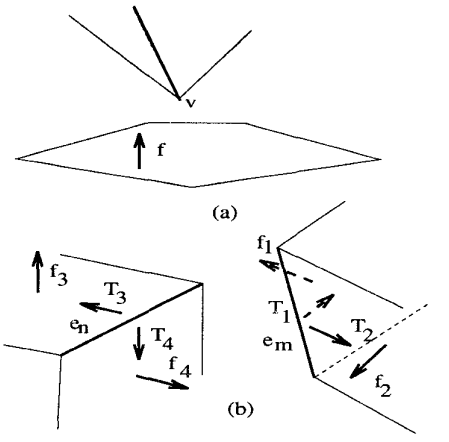

Figure 1: *(a) Applicable vertex - face pairing. (b) Applicable edge - edge pairing.*

By applying these criteria, the number of candidate pairings is restricted considerably. On the one hand, as mentioned before, following the vertex - face applicability criterion, only one edge has to be considered for every face, leading to a linear number of candidate edge-face pairings. As for candidate pairings arising from the edge - edge applicability constraints, a worst case can be found with a quadratic number of candidate edge - edge applicable pairs, therefore leading to a quadratic number of candidate edge - face pairings. The worst case arises in geometries where, in both polyhedra, a set of edges approaches a circumference and the cardinality of this set is of the same order as the complexity of each polyhedron. This happens for pyramids or bipyramids whose apices are vertices of high degree, and where their height (compared to the base) is small and/or their relative orientation is close to perpendicularity. It happens also for two prisms whose respective bases have a large number of sides, for any orientation (except perfect parallelism). Nevertheless, for most convex polyhedra the number of candidate pairings is strongly subquadratic and approaches linearity, as confirmed by the results obtained in Section 5.2.

[1] Every other contact between the features of two polyhedra can be expressed in terms of these basic contacts.

As mentioned before, applicability constraints were originally developed for convex polyhedra. If the polyhedra are non-convex, these constraints express a necessary but not sufficient condition for contact. It makes sense to talk about *local* applicability, as far as only the adjacent features are considered in the applicability constraints, but other features of the polyhedra can keep the locally applicable contact from being actually possible. In other words, size plays now a role and it is not considered in the applicability constraints. Figure 2 shows how a locally applicable contact cannot be realized. Candidates arising from such a situation will be called *false* candidates.

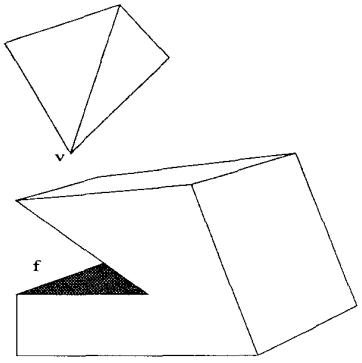

Figure 2: *A locally applicable vertex - face contact that cannot be realized.*

On the other hand, as they are necessary conditions, applicability constraints can still be used in the context of a geometric pruning strategy: if they do not hold, the corresponding pairing can be discarded in the search of edge - face candidates for intersection. Of course, this strategy is now conservative, in the sense that false candidates may arise. A worst case can be found where the number of false candidate pairings is quadratic, but it is attached to a very specific and unfortunate geometry (as, for example, the situation depicted in Figure 3).

Some comments about features that can be locally applicable in the case of non-convex polyhedra: In vertex - face contacts, no restriction exists about the faces. As for vertices, they can be only of two types: convex vertices or mixed ones that have a *local* convex hull (*pseudo-convex* vertices). In Figure 4 a pseudo-convex vertex and its local convex hull are shown. Clearly, in edge - edge contacts, edges must be convex.

Now the issue is to find an efficient way of obtaining the vertex - face and edge - edge pairs satisfying the applicability constraints, without exploring explicitly every vertex - face and edge - edge pair. The resulting algorithm should be of the same complexity as its output. The key is to make use of an adequate representation.

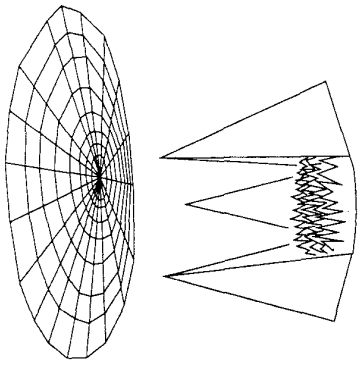

Figure 3: *The number of false edge - face pairings is quadratic.*

4 The Spherical Face Orientation Graph (SFOG)

Several authors have developed spherical representations of convex polyhedra. In [11] a detailed description of *Extended Gaussian Images* (EGI) and their properties can be found. In the case of polyhedra, these images represent face orientations as points on the unit sphere, and each point is weighted according to the area of the face it represents.

A representation that explicitly captures both geometrical and topological information is desired. This can be done with the **Spherical Face Orientation Graph (SFOG)**, which is particularly well suited for exploring the applicability constraints in an efficient way. This representation is inspired by the EGI, but it does not associate weights proportional to the area of each face with each point (node) on the sphere. Instead, topological relations of adjacency between faces are explicitly depicted, with arcs joining nodes that correspond to faces sharing an edge. Geometric consistency is attained if these arcs are not arbitrary but lie on great circles of the sphere[2]. Convex edges are represented by means of the minor arc, concave edges with the major arc. The resulting spherical graph is not an unambiguous representation of polyhedra, or, in other words, a polyhedron may not be reconstructed from this representation, neither in size nor completely in shape (for example, any rectangular prism has the same SFOG representation), but it preserves those geometric relations that are

[2]The normals of the planes that define these great circles point in the same directions as the corresponding edges.

relevant to the applicability constraints.

A vertex is represented by means of a cycle of arcs and nodes, corresponding to the adjacent edges and faces. If the vertex is convex or pseudo-convex, there exists always a subset of convex arcs that bound a convex polygonal region on the sphere. For pseudo-convex vertices several such regions may exist, but only one is contained in every other one. This smallest region represents the local convex hull of the vertex, and will be called *convex subregion (csr)*. Figure 4 shows a pseudo-convex vertex, its local convex hull, and the corresponding *csr* on the SFOG.

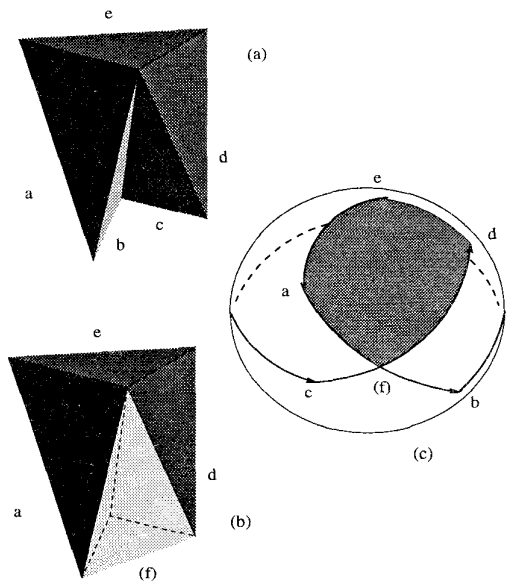

Figure 4: *(a) A pseudo-convex vertex, (b) its local convex hull, and (c) the corresponding* csr.

By superimposing the SFOG of one polyhedron with the central symmetric image of the SFOG of another polyhedron, a compact representation is obtained from which the vertex-face and edge-edge applicability relationships can be directly determined:

1. (Convex case) A given node falls into a certain region if and only if the contact between the vertex represented by the region and the face represented by the node is applicable (Fig. 5(a)).
2. (Non-convex case) A given node falls into a certain convex subregion if and only if the contact between the vertex whose local convex hull is represented by the *csr* and the face represented by the node is *locally* applicable.
3. Two convex arcs of different SFOGs intersect if and only if the contact between the corresponding edges is (*locally*, in the non-convex case) applicable (Fig. 5(f)).

These results can be easily proved by taking the geometrical correspondence between arcs and edges into account. For example, point 1 above can be proved in the following way:

Consider the vertex v -face f applicability constraint (Type A contact in Section 3). The interior of a convex region on a sphere, formed by arcs of great circles, is well defined: a point f lies in the interior of the region v if, travelling counterclockwise along the perimeter, f is on the left of every arc $q_m - q_n$. This condition is expressed by the triple product $\langle q_m \times q_n, f \rangle \geq 0$. The product $q_m \times q_n$ defines the direction of the corresponding edge. The arc $q_m - q_n$ is common to the region that represents v and the region that represents an adjacent vertex, say v_k. In other words, it represents the edge from v_k to v and, therefore, $q_m \times q_n$ has the same direction as $v - v_k$. Note that as one of the SFOGs has been inverted, this implies that $\langle v_k - v, f \rangle \geq 0$. Thus, the point-in-region inclusion condition is equivalent to the vertex-face applicability condition.

As for edge-edge applicability, the corresponding result can be proved in a similar fashion, projecting the directions of the tangent vectors (see Figure 1) onto f_p, whose direction is given by any one of the intersection points of the great circles that contain the arcs.

The previous results are consistent with several facts concerning the applicability constraints (valid both in the convex and non-convex case):

- Each SFOG defines a set of regions that covers the sphere. Every node from one SFOG will be contained into one region from the other or in its border (it may fall on an arc or on a node). This coincides with the fact that for every face of one polyhedron there will be (at least) one applicable vertex of the other polyhedron, as shown in Fig. 5(a).
- It may be that for a given region there is no node of the other SFOG that falls into it. This is coherent with the fact that there can be vertices which are not applicable with respect to any face of the other polyhedron (see Fig. 5(b)).
- It is also possible that two or more nodes are contained in the same region. In fact, it is possible that the same vertex be applicable to several faces (see Fig. 5(c)).
- An arc can be intersected by one or several arcs of the other SFOG, which means that one edge

can be applicable to more than one edge of the other polyhedron. It can also happen that no arc intersects a given arc, i.e. there is no applicable edge with respect to the edge represented by this arc.

- Degenerate situations find their counterparts in the representation:
  - "A node falls on an arc" means that the edge represented by this arc is parallel to the face corresponding to the node (see Fig. 5(e)).
  - "Two arcs are on the same maximal circle" means that the represented edges are parallel (see also Fig. 5(e)).
  - "A node coincides with a node" means that the two faces (each belonging to a different polyhedron) are parallel (see Fig. 5(d)).

If the polyhedra are non-convex, another fact has to be considered:

- Convex subregions may overlap. If a given node is contained in one of these common areas, the represented face will be simultaneously locally applicable with respect to the corresponding vertices.

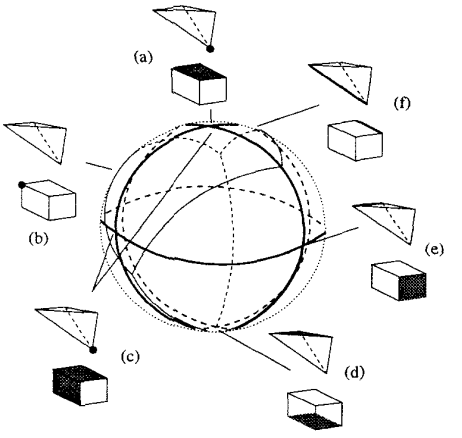

Figure 5: *The SFOG of a rectangular prism (heavy lines) is combined with the central symmetric image of the SFOG of a tetrahedron (fine lines). Different situations are represented: (a) applicable vertex-face contact, (b) a vertex of the prism which is not applicable to any face of the tetrahedron (no node of the tetrahedron is inside the corresponding region), (c) a vertex of the tetrahedron which is applicable with respect to two faces of the prism simultaneously, (d) parallel faces, (e) an edge which is parallel to a face, (f) applicable contact between edges.*

5 Searching for candidate pairs

The next step is to develop an algorithm that determines the applicable vertex-face and edge-edge pairings, using the information contained in the SFOGs. Once these applicable pairings have been obtained, it is straightforward to obtain the candidate edge-face pairs, as described in Section 3.

First, in Section 5.1, a general algorithm will be sketched, that applies for convex as well as for non-convex polyhedra. The amount of pruning that can be done becomes particularly evident in the convex case. Therefore, a simple algorithm has been developed and implemented for the situations where the polyhedra are known to be convex, as shown in Section 5.2, and experimental results are also provided.

5.1 The general case

For non-convex polyhedra, convex subregions have to be identified and arc crossings of two kinds (between convex arcs of the same SFOG and between convex arcs of different SFOGs) have to be distinguished. The algorithm has to perform three main tasks: To detect arc crossings, to detect regions overlap, and to detect node-in-region inclusions. At the base lies the arc crossings detection procedure. It consists in applying a modified version of segment intersection detection algorithms through line sweeping, like that in [12] ($O(n+k)\log n$) or in [13] ($O(k+n\log n)$, for n segments and k intersections). This algorithm has to be adapted for treating arcs on the sphere instead of segments in the plane, as well as for distinguishing between arcs of the same and of the other SFOG. The sweep with a vertical line is replaced by a sweep with a meridian. The partial ordering that this sweep induces on the arcs is used to keep track of the regions that are being swept and this, in turn, allows to perform the other two tasks of region overlap and node-in-region inclusion detection. In fact, these two steps are merged together, as far as the aim of the region overlay detection step is to allow knowing in which regions a given node lies, that is, which vertices are simultaneously applicable with respect to the same face.

5.2 The convex case

The algorithm shown below considers nodes of one SFOG and regions of the other one, and performs the node-in-region inclusion and the arc crossings

tests. Obviously, complexity will not be increased by applying further the same algorithm to the nodes of the second SFOG and regions of the first one.

As for data structures needed in the algorithm, there are input graphs, the SFOGs and a Cycle Graph (where nodes correspond to vertices of one polyhedron), a vector FACE_APP[<node>] recording the face-vertex applicability relationships and a vector of lists EDGE_APP[<arc>] recording the edge-edge applicability relationships, which are also the desired outputs, and two commonly used lists in these search algorithms [14], OPEN_NODES and OPEN_ARCS.

A brief description of procedures and functions is needed for the clear understanding of the algorithm. The procedure *ordered_intersection (node, &EDGE_APP)* finds every intersection of arcs stemming from *node* with the arcs of the cycle in which *node* lies, and appends these intersections to EGDE_APP. The *arc_intersection(arc, cycle, edge)* function finds the intersection between *arc* and the arcs of *cycle* different from *edge*, which is known to have been crossed by *arc* in entering *cycle*. The function *Succ_Arcs(node)* returns the arcs that "point out of" *node*, in the sense that although we are exploring an undirected graph, certain directions of the arcs are implicitly imposed as some nodes are explored before others and we want to avoid exploring a given arc in both directions. The function *succ_node(arc)* returns the unexplored extreme node of *arc*, and *succ_cycle(edge, arc)* returns the cycle that cobounds *edge* and where *arc* is "pointing to" (in the sense that the other cobounding cycle will either contain the node such that $arc \in Succ\_Arcs(node)$ or will already have an arc intersected by *arc*). Finally, the function *last(EDGE_APP[arc])* returns the last arc intersected by *arc*.

The algorithm starts at a given point, which can be considered without loss of generality as a "North Pole", and travels over the sphere towards the "South" in a spiral-like fashion. Thus, it can be considered a greedy or breadth-first algorithm [14].

SFOG SEARCH ALGORITHM

```
Choose North-pole;
Find North-region ⊃ North-pole;
FACE_APP[North-pole]:= North-region;
North-pole → OPEN_NODES;
while (OPEN_NODES ≠ ∅) or (OPEN_ARCS ≠ ∅)
    while (OPEN_NODES ≠ ∅)
      node ← OPEN_NODES;
      ordered_intersection(node, &EDGE_APP);
      for every a ∈ Succ_Arcs(node)
        if EDGE_APP[a] = ∅ then
          if FACE_APP[succ_node(a)]= ∅ then
            FACE_APP[succ_node(a)]:=
                FACE_APP[node];
            succ_node(a) → OPEN_NODES;
          endif
        else
          a → OPEN_ARCS;
        endif
      endfor
    endwhile
    while (OPEN_ARCS ≠ ∅)
      arc ← OPEN_ARCS;
      edge := last(EDGE_APP[arc]);
      cycle := succ_cycle(edge, arc);
      s := arc_intersection(arc, cycle, edge);
      if s=∅ then
        if FACE_APP[succ_node(arc)] = ∅ then
          FACE_APP[succ_node(arc)] := cycle;
          succ_node(arc) → OPEN_NODES;
        endif
      else
        EDGE_APP[arc] ← s;
        arc → OPEN_ARCS;
      endif
    endwhile
endwhile
```

The algorithm has been implemented and experiments have been carried out, consisting in the execution of the algorithm on pairs of polyhedra with a given relative orientation. The set of polyhedra used in the experiments covers the range from the tetrahedron to a polyhedral approximation of the sphere with 128 triangular faces. As can be seen in Figure 6, the number of edge-face candidates found in the experiments grows linearly with the complexity of the 34 pairs of convex polyhedra considered. The two points that lie clearly apart from the line correspond to the exceptional situations described in Section 3, bipyramids and prisms with a large number of sides.

6 Conclusions and further research

Most known interference detection algorithms [1,8, 9], which have been mentioned in the introduction, take advantage of the convexity of the polyhedra to attain efficiency. Here, a general algorithm has been described, which can be applied to convex as well as to non-convex polyhedra, without the need of a previous decomposition step.

The edge - face intersection test for interference detection between polyhedra is simple and easy to implement, and does not need sophisticated data

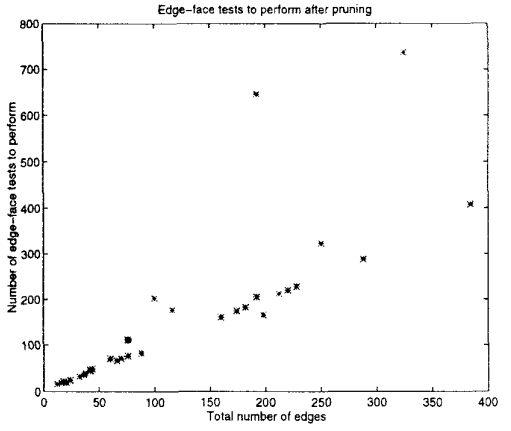

Figure 6: *Experimental results show a linear relationship between the number of elementary edge-face tests that have to be performed and the total number of edges. Furthermore, the constant of linearity is close to 1.*

structures. Algorithms based on this paradigm have a quadratic worst-case complexity. Nevertheless, it has been shown that, using geometric pruning techniques based on the applicability constraints, an expected computation time that grows linearly with the complexity of the polyhedra can be attained, if they are convex. Note that this is the same expected running time provided by algorithms designed specifically for the convex case [8,9]. In the non-convex case the expected complexity is also subquadratic. The number of false candidates that will be tested for interference depends not only on the degree of non-convexity but also on the relative sizes of the polyhedra.

A representation that captures the applicability relations between the features of two polyhedra, which we call SFOG, has been presented. It is particularly well suited for the convex case, where an algorithm based on this representation has been implemented and results have been obtained that show its good performance in the average.

Further work includes three main directions. First, the general algorithm described in Section 5.1 has to be implemented, in order to obtain experimental results for the non-convex case. Second, rotation may be considered, where the same representation can be used to divide the rotational motion into intervals where applicability constraints remain constant. Finally, the obtained algorithms have to be integrated in a collision detection scheme.

7 References

[1] D. Dobkin and D. Kirkpatrick, "Determining the separation of preprocessed polyhedra -a unified approach," *ICALP-90*, 443, pp. 400–413, 1990.

[2] K. Mehlhorn and K. Simon, "Intersecting two polyhedra one of which is convex," *Fundamentals of Computation Theory 85, Lecture Notes in Computer Science*, 199, pp. 534–542, 1985.

[3] C. Bajaj and T. Dey, "Convex decomposition of polyhedra and robustness," *SIAM J. Comput.*, 21, no. 2, pp. 339–364, Apr., 1992.

[4] F. Thomas and C. Torras, "Interference detection between non-convex polyhedra revisited with a practical aim," *IEEE Proc. Int. Conf. Robotics Automat.*, 1, pp. 587–594, May, 1994.

[5] J. W. Boyse, "Interference detection among solids and surfaces," *Comm. ACM*, 22, no. 1, pp. 3–9, Jan., 1979.

[6] T. Lozano-Pérez, "Spatial planning: a configuration space approach," *IEEE Trans. Comput.*, 32, no. 2, pp. 108–120, Feb., 1983.

[7] J. F. Canny, "The complexity of robot motion planning," The MIT Press, Cambridge (MA), PhD Thesis, 1988.

[8] E. G. Gilbert, D. W. Johnson and S. Keerthi, "A fast procedure for computing the distance between complex objects in three dimensional space," *IEEE J. Robotics Automat.*, 4, no. 2, pp. 193–203, Apr., 1988.

[9] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," *IEEE Proc. Int. Conf. Robotics Automat.*, 2, pp. 1008–1014, Apr., 1991.

[10] B. R. Donald, "Local and global techniques for motion planning," Massachusetts Institute of Technology, Masters Thesis, 1984.

[11] B. K. P. Horn, "Extended Gaussian Images," *Proc. of the IEEE*, 72, no. 12, pp. 1671–1686, Dec., 1984.

[12] J. L. Bentley and T. A. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Comput.*, 28, no. 9, pp. 643–647, Sept., 1979.

[13] B. Chazelle and H. Edelsbrunner, "An optimal algorithm for intersecting line segments in the plane," *Journal of the ACM*, 39, no. 1, pp. 1–54, Jan., 1992.

[14] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.